EV369763495

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

of

Eric D. Deily
Ming Lu
Gabriele Giuseppini
Melur Raghuraman

and

Jaroslav Dunajsky

for

Tracing A Web Request Through A Web Server

ATTORNEY'S DOCKET NO. MS1-1906US

Tracing a Web Request through A Web Server

TECHNICAL FIELD

[0002] The present invention relates generally to event tracing and more particularly to a method of tracing a Web request through a Web server.

BACKGROUND

[0003] In the client/server World Wide Web of Internet computing that we live in today, increasing numbers of Web based applications are being developed – an example of which is electronic commerce (e-commerce) or 'business on the Web'. A Web based application includes the functionality of a Web server servicing a request received through the Web from a client (i.e., a Web request). The development of Web based applications includes the task of finding errors in programming code, often referred to as 'debugging' a computer program. Errors in programming code can also show up when a Web server services a Web request received through the Web from a client.

[0004] Common issues in debugging Web applications include finding out what is happening with a client system and what is happening with a Web application running on the client system. In order to diagnose issues on servers and clients, such as performance on a Web request and/or issues of a failed Web request, no easy way exists to figure out what's going on while a Web request is being processed. One way to diagnose such issues is to trigger a dump of the right processes at the right time. Then, after the often quite large dump

has been retrieved, a review of the contents of one or more files of the dump is needed to find one or more sources of a poor request performance or of a Web request failure. This dump review process can require highly skilled diagnostic personnel in order to be performed properly and it may be quite difficult for skilled diagnostic personnel to find the source of the failed Web request. Moreover, evidence of a poor request performance or of a Web request failure might not even be in the dump since more often than not the dump will be triggered at a time when the client and the server are properly operating.

[0005] Another common problem in debugging Web requests is figuring out where performance issues arise on specific kinds of Web requests. For instance, diagnosis that is global across an entire server so as to hit all Web requests is not granular enough to be practical. If a particular kind of Web request has severe latency problems and that Web request involves a plurality of installed filters, it may be necessary to review a dump reporting on all of the filters in order to find out that a problem is arising with only one of the installed filters. Alternatively, it may be necessary to enable one filter at a time and perform successive individual tests for each of the installed filters to find out if a problem is arising with only one of the installed filters. These types of problem resolution efforts can be time consuming.

[0006] Still another common problem in debugging Web requests is figuring out why a specific Web request stops being serviced (e.g., 'hangs'). Again, this may require the triggering of a dump and reviewing content of the dump files, where the dump was hopefully triggered at an appropriate time to include evidence of Web request servicing errors. Even then, the right amount of information in the dump files may be insufficient for a proper diagnosis or may be too verbose to be practical for debugging any one of several Web requests that are reflected in the dump files. Yet another problem with debugging

some Web requests is that, in order to perform a proper debugging analysis, the specific Web request must not 'hang' but must continue on to completion of the Web request before all of the events for that Web request are stored and accessible for diagnostic analysis.

[0007] One source of problems in debugging Web request problems is that operating personnel may just cancel a Web request hosting process when a Web request servicing problem occurs. Unfortunately, this can remove any trace or evidence needed for a proper diagnosis of the Web request servicing problem.

[0008] Another type of difficulty in Web request servicing problem diagnosis is when code does not stop executing as soon as a bug occurs and the bug occurs without any error being signaled. As a result, the 'bug' is not noticed until long after the bug has occurred. Then, diagnostic personnel must undertake the often difficult task of finding how and where the bug occurred during execution of the code – such as by sifting through a voluminous amount of data in one or more dump files.

[0009] It would be an advance in the art to document (e.g., create a 'trace' for) a Web request as it proceeds throughout an entire servicing of the Web request, or a specifically identified portion thereof, by a Web server. The trace can then be used to easily identify one or more sources of a Web request servicing problem - thereby increasing the probability of having an easily reproducible test case that can be used in debugging by a software developer or by diagnostic personnel.

SUMMARY

[0010] In one implementation, a demand can be made for a trace of identified key execution events as they happen for a Web request that being serviced by a Web server. These events can be correlated to a unique Web request identifier (ID) for each Web request. Thus, every happening or event that is traced during the execution of a Web request will be associated

with the unique Web request ID. This unique Web request ID can be positioned, for convenience and ease of reference, so as to be the first piece of user data in a trace entry or record that is associated with the Web request. An application can then use the unique Web request ID to access a log file that contains all entries or records that trace all of the events that happen during the servicing of the Web request. This access to the trace in the log file allows the application to produce a report on the identified key execution events that happened for the Web request as the Web request makes its way through the Web server.

[0011] In other implementations, an application program interface is exposed for Web based applications that make Web requests, such as electronic commercial (e-commerce) applications. These Web based applications can then obtain the unique Web request ID. Once obtained, a Web based application can correlate its own events with the events of the Web server in servicing Web requests from the Web based application.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] A more complete understanding of the implementations may be had by reference to the following detailed description when taken in conjunction with the accompanying drawings wherein:

[0013] Figure 1 illustrates a network architecture in which clients access Web services provided by one of more servers over the Internet using conventional protocols, where each server runs a server process that can access one or more databases;

[0014] Figure 2 illustrates an exemplary embodiment of an event trace output file, or a event trace log file;

[0015] Figure 3 is a flowchart of an exemplary process for logging events that happen into an event trace output file during the servicing of a Web request;

[0016] Figure 4 is a flowchart of an exemplary process for producing a demand report of events that happen during the servicing of a Web request, where a trace log file is opened to obtain an identifier for each trace event, where each identifier for each trace event is used to decode the data in each trace event, and where Web request identifiers are also used to uniquely identify each Web request and all of the trace events that correspond to the Web request.

[0017] Figure 5 is a block diagram of an exemplary environment capable of supporting any server or client depicted in Figure 1.

[0018] The same numbers are used throughout the disclosure and figures to reference like components and features. Series 100 numbers refer to features originally found in Fig. 1, series 200 numbers refer to features originally found in Fig. 2, and series 300 numbers refer to features originally found in Fig. 3, and so on.

DETAILED DESCRIPTION

[0019] A Web request event tracing tool, in various implementations, provides an environment for a system to diagnose problems with Web request processing of Web applications. A tracing mechanism is provided by the tool that allows for all execution events that happen during the processing of a Web request to be tied back together for that single Web request. The events that are tracked for a Web request can be those that occur through a unit of work, an activity, or any other kind of recognized event that can happen.

[0020] An operating system for a server can include the Web request event tracing tool. The tool can be maintained by the operating system, where the operating system has a trace logger that can accept data from both kernel mode device drivers and user mode applications. All functionality of the Web request event tracing tool, however, need not be

provided by the operating system but can also be provided, either in whole or in part, by an application that works in conjunction with the operating system of the server.

[0021] A network environment 100 is depicted in Figure 1. Network environment 100 includes a plurality of different kinds of clients 120 (1-M) that access Web services provided by one of more servers 134 over the Internet using conventional protocols. Each server 134 runs a server process that can access one or more databases 114. Each server 134 can operate a Web request event tracing tool via its operating system or, either in whole or in part, via one or more server applications 130 that work in conjunction with the operating system of the server 134.

[0022] The network environment 100 includes representative Web services accessible directly by a software application, such as Web application 110. Each Web service is illustrated as including one or more servers 134 that execute software to handle requests for particular services. Such services often maintain databases 114 that store information to be served back to requesters. For instance, databases 114 can include an object-oriented database. Web services may be configured to perform any one of a variety of different services and can be combined with each other and with other applications to build intelligent interactive experiences.

[0023] The network environment 100 also includes representative client devices 120(1), 120(2), 120(3), ..., 120(M) that utilize the Web application 110 (as represented by communication links 122-128). The client devices, referenced generally as number 120, can be implemented many different ways. Examples of possible client implementations include, without limitation, portable computers, stationary computers, tablet PCs, televisions/set-top boxes, wireless communication devices such as cellular telephones, personal digital assistants, video gaming consoles, printers, photocopiers, and other smart devices.

[0024] The Web application 110 is an application designed to handle and service requests from clients 120. The Web application 110 is composed of one or more server applications 130 that are executing on one or more servers 134 or other computer systems. A portion of Web application 110 may actually reside on one or more of clients 120. Alternatively, Web application 110 may coordinate with other software on clients 120 to actually accomplish its tasks.

[0025] Each server 134 has an operating system 142. Each operating system can have a trace infrastructure 144 that includes a Web request event tracing tool. Alternatively, the Web request event tracing tool can also be provided, either in whole or in part, by a Web request trace application 140 that works in conjunction with the operating system 142 of the server 134.

[0026] The operating system 142 of each server 134 has an application program interface (API) layer 146 and an operating system/services layer 148. The API layer 146 presents groups of functions that the server applications 130 can call to access the resources and services provided by layer 146. An application residing on client 120 can also use the API functions by making calls directly, or indirectly, to the API layer 146 over the network 104.

[0027] As part of the functionality provided by the Web request event tracing tool, a "trace context" is obtained for a Web request. The tool then uses that trace context in the events that happened during the servicing of the Web request so as to be able to correlate trace events from one or more applications that interface with a server that services the Web request as well as with events in the server. By way of example, an exemplary trace context for a Web request is a Globally Unique Identifier (GUID) that is associated with the Web request. Each event that happens during the servicing of the Web request is then associated with the Web request GUID. As the event happens, the event is logged with the Web

request GUID in a log file that traces the servicing of the Web request. Thus, every event that happens when the Web request is serviced is traced during the execution of the Web request so as to be associated with the Web request GUID. The Web request GUID is used in implementations of the Web request event tracing tool to track the events for a Web request as it is serviced both at a server as well as when the server accesses additional information, such as by way of a database access. In order to be unique the Web request GUID can be constructed, for instance, as a conventional 128 bit GUID.

[0028] Each entry in the log file for each event can include, in addition to the Web request GUID, a variety of other information. This information can be specific to the type of event that is being logged. For instance, the entry can include the kernel mode time, the user mode processor time, a process identifier (ID), a thread ID, etc. Each process incident to servicing a Web request can be logged with various information: an image filename that the process was running; a Web request send; etc. The Web request GUID can be positioned, for convenience and ease of use, so as to be the first piece of user data in the trace that is associated with the Web request in a log file.

[0029] An exemplary embodiment of an event trace output file (e.g., an event trace log file) 202 is illustrated in Figure 2. File 202 can be one or more event trace log files in an environment 200. For instance environment 200 can be within the one or more databases 114 seen in Figure 1, where each server 134 runs a server process that can access the one or more databases 114 for the purpose of interacting with the one or more event trace log files 202. Note that the databases 114 can be remote from any server 134 and that trace events can be published to a remote location by the server 134.

[0030] Event trace log file 202 can contain a plurality of entries. Each entry represents an event 204 (k), where k is an integer from 1 to K. Event 204 (k) is logged into trace log file

202 as it occurs during the processing of a Web request by server 134. A Web request GUID 206 is associated with the Web request by a trace infrastructure, such as by Web request trace application 148 or by trace infrastructure 144. Each event 204(k) is logged with the Web request GUID 206 that corresponds to the Web request within which the event occurred. Along with the Web request GUID 206 for the particular Web request, one or more data types 208 are included with each entry in the log file 202. Each data type 208 can include a different kind of data that can be descriptive of that particular event 204(k) that occurred during the processing of the Web request to which the Web request GUID 206 has been assigned. The number of data types 208 for an event 204(k) can depend upon a specified level of verbosity that is to be logged for the particular event. Each event 204 has a corresponding 'event GUID', for instance, Data Type (1) 208. As such, there is one (1) event GUID 208 for each event 204 and there is one (1) Web request GUID 206 for each Web request.

[0031] A consumer application can use the unique Web request ID (e.g., the GUID) to access the first piece of user data in the trace within the log file which, as previously described, is where the Web request GUID can be stored. Since the log file contains the trace of all the events that happened for the Web request, this access to the trace in the log file by the application allows the application to produce a report on identified key execution events that happened as the Web request makes its way through one or more processes that are performed as the Web request is being serviced by the Web server. The report is based upon the Web request GUID that is associated with the Web request. The application that generates the report uses the Web request GUID to put back together or reconstruct which events happened as those events are reflected in a trace log file for a traced process.

[0032] In an exemplary Web request server environment, multiple logger streams may be active at one time, typically one for a kernel logger and one for each trace-enabled application that is running on the server. The trace from multiple logger streams can be processed and returned to a caller one event at a time. By way of example, the caller can be a trace-enabled application that is running on the server. The application can access the logger streams to a trace for a Web request that is being logged in a log file. The application can retrieve information about the servicing of a Web request from the log file and can then use this information to produce a report. The report can be used to find a source of a problem that is related to an event that happened during the servicing of a Web request.

[0033] The functionality of the Web request event tracing tool can be applied to any Web based application. The Web based application can make use of an event tracing mechanism to do Web request tracing which is available to all tracing entities. The event tracing mechanism generates traces events for events that happen during the course of servicing of a Web request. The functionality of the Web request event tracing tool can obtain and/or generate a Web request GUID for a Web request. The tool then associates the Web request GUID with events that happen during the servicing of the Web request. As events happen during the servicing of the Web request, an entry can be made for each event in a log file. Note also that each event has an event GUID that is entered into the entry for that event in the log file. Each entry corresponding to the servicing of the Web request that is logged in the log file will be associated with the Web request GUID. A trace-enabled application that is running on the server that is servicing the Web request can then use the Web request GUID to access the log file. Alternatively, events in the log file can be sorted or otherwise grouped according to their respective Web request GUIDs in order to find out what events were logged for each Web request. In either case, a diagnosis can be made of problems that

happened when the Web request was being serviced. This diagnosis finds a correlation between the trace events that occur when the Web application interfaces with the server and the trace events that occur within the server itself as it services the Web request. As such, implementations provide diagnosability when problems happen during the processing of the Web request. Such implementations thereby obviate the need to create or generate a process dump for review and stack analysis when a problem happens during the processing of the Web request. This diagnosability is provided by implementations of request-based tracing where a Web request GUID (i.e., the trace context ID) is flowed across to the Web application this is interfacing with the server that services that application's Web request. This Web request GUID allows the Web request to be watched as it makes its way through the system and back out.

[0034] While all events that happen during the course of servicing a Web request can be traced, not all events need to be traced. Rather, implementations of the Web request-based tracing tool can be directed to trace only specific Universal Resource Locator's (URL's), as well as to trace only certain areas of functionality. As such, an administrator can choose what URL's they want to trace so that not every URL is traced. Moreover, rather than tracing all Web requests, only those Web requests that are needed to be traced are traced.

[0035] Implementations of Web request-based tracing tool can limit tracing too only certain functionalities. For instance, tracing can be limited while a Web request is being serviced to the functionalities of authentication, security, compression, Common Gateway Interface (CGI), and/or filters. This limits tracing to only desired events, thus decreasing the amount of events raised and reducing the noise and the overall effect on system resources (disk, CPU, etc.).

[0036] In the case where Web request-based tracing is limited to a particular filter of interest, a trace can be put around a call into the filter. The trace log would then show what is going into the filter as well as what is coming out of the filter. A trace can be set up to only trace filter calls.

[0037] Yet another limitation can be placed upon the Web request event tracing tool with respect to the amount of information that is logged for each event that happens. This limitation is referred to herein as the level of 'verbosity' for the events that are desired to be logged. As discussed above with respect to Figure 2, one or more data types 208 can be descriptive of a particular event 204(k) that occurred during the processing of the Web request to which the Web request GUID 206 has been assigned. The number of data types 208 for an event 204(k) can depend upon a specified level of verbosity that is to be logged for the particular event 204(k). The level of verbosity can be, for example, set to a certain threshold as follows:

> 0 - No Data: The trace log will include only those events that give data that is required for a proper context as to what is occurring as a Web request is being serviced.
>
> 1 – Fatal: The trace log will include '0', above, and an abnormal exit or termination; If an action will cause a process to exit, or if the problem can be caught before the process terminates, then the event is logged as a Fatal Error. As such, the Web request event tracing tool can persist in tracing events through an unexpected process termination.
>
> 2 – Error: The trace log will include the above and severe errors that should be logged; If a process can not continue and the processing must immediately reject the Web request, and then the event is logged as an Error. (e.g., Anonymous user request/account password expired/password expires on account).
>
> 3 – Warning: The trace log will include the above and the logging of the events that are warnings such as a resource allocation failure; if an unexpected result/error occurs but processing of the Web request can continue, the event will be logged as a Warning. (e.g., unhealthy server conditions; basic authentication failure; login failure).
>
> 4 – Information: The trace log will include the above and non-error events such as Entry-Exit sequences or a process and other informative events.
>
> 5 – Verbose: The trace log will include detailed traces from intermediate steps that occur during the servicing of a Web request.

[0038] The Web request event tracing tool can be activated at a server by an administrator. This activation instructs each worker process at the server to start tracing requests. If URL filtering is enabled, the server will check the URL and determine whether or not it is one of the previously specified URL's to trace. As the Web request is serviced by the server, the server determines whether or not it should publish a trace event for the request in key areas based on the functionality, verbosity, URL filtering rules, etc. Each event can publish the request ID (e.g., the Web request GUID) as the first part of user data in the trace event that is logged in a trace log file. In particular, the server's Web request-based trace events can be published with the Web request GUID as the first part of the user data published in each trace event in the log file. This way, an application can access the log file using the Web request GUID and can then correlate all the trace events that happened during the servicing of the Web request.

[0039] In one exemplary implementation where the Web request event tracing tool is a component of a server's operating system, Application Program Interfaces (APIs) are provided. These APIs allow Web applications that are running on or interfacing with the server to publish their own events as they happen, where these events are related to a Web request. These events are then logged with the associated Web request GUID that corresponds to each Web request that is being serviced by the server. In another exemplary implementation, the Web request event tracing tool can be turned on and off from remote locations, such as by various telecommunication tools. Additionally, trace events can be published to a remote location.

[0040] The Web request event tracing tool can be implemented in various way. One such implementation uses the tool to track a latency troubled Web request that is being serviced in order to find out where the latency is occurring. For instance, tracing can be used to track

detailed events for long running Web requests, such as key transitions that occur when the server interfaces with Web based applications. In another implementation of the tool, the tool is able to get trace data even if a process crashes when a Web request is being serviced. Some times data for a Web request isn't put into an event log because code that is executing crashes and the writing to the event log does not take place for a Web request until after the Web response is determined and sent. Thus, if a particular Web request is crashing, the Web requests are not seen in the event log. Therefore, the tool is configured to ensure that trace data is persisted even through a process crash. Then, the Web request GUID corresponding to the Web request can be used to access the log file of events. The events corresponding to the Web request GUID can be viewed for each process that occurred as the Web request was being services. Moreover, the review of trace data can be remotely managed and reviewed.

[0041] An exemplary configuration of the Web request event tracing tool is to trace events so as to capture only information on requests that have "failed". This might be done because it may be difficult to reproduce circumstances under which a process fails.

[0042] As discussed above, the functionality of the Web request event tracing tool can be applied to any Web based application. For purposes of illustration, and not for limitation, one common category of a Web based application is an electronic commerce (e-commerce) application. An exemplary e-commerce application provided by the Microsoft Corporation of Redmond, Washington, includes Internet Server Application Program Interface (ISAPI) extensions and an Internet Information Services Server (IIS). An ISAPI extension implements API for the IIS and can be used to extend the server functionality. Examples of this functionality include ASP and ASP.NET provided by the Microsoft Corporation. ASP implements an interface into which IIS can make calls. As such, examples of ISAPI

extensions can include ASP and ASP.NET. ISAPI is an interface layer between IIS and extensions that extend the functionality of the server.

[0043] The IIS can make use of an Event Tracing for Windows® (ETW) infrastructure, which is also provided by the Microsoft Corporation. ETW can do Web request tracing and is available to all tracing entities. In this exemplary e-commerce application, IIS and ISAPI calls into ETW to generate trace events which ETW then buffers and flushes to disk. The functionality of the Web request event tracing tool can be used in this exemplary Microsoft e-commerce application where the tool obtains and/or generates a Web request GUID for an e-commerce related Web request. For instance, all existing ETW events and new IIS trace events can use the Web request GUID as a first data item in an entry in a log file. In this environment, the server can have an executable that controls the tracing infrastructure that turns ETW on and off (i.e., logman.exe). The executable can call into the tracing infrastructure and set up a tracing session internally, and can provide a session handle to each process that is to run during the servicing of a Web request be the server.

[0044] The tool then associates the Web request GUID with events that happen during the servicing of the Web request. As events happen during the servicing of the Web request, an entry can be made for each event that is of interest in a log file, along with a specified level of verbosity. Each entry in the log file will be associated with the Web request GUID. A trace-enabled application that is running on the server that is servicing the Web request can then use the GUID to access the log file. This access allows a diagnosis to be made of the problems that happened when servicing the Web request. Thus, by using the Web request GUID associated with the Web request in assessing the trace in the log file, a correlation can be found between the ISAPI's trace events and the IIS trace events.

[0045] Implementations provide diagnosability of Web request-based tracing where a Web request GUID is flowed across to ISAPI extension layers of Web request processing. The ISAPI extension layers can include a component of the operating system and/or middleware. This Web request GUID allows the Web request to be watched as it makes its way through the system and back out. ASP.NET (e.g., ASPNET.dll) and ASP (e.g., ASP.dll) are both examples of ISAPI extensions that are loaded into a worker process. These ISAPI extensions call into IIS via ISAPI API's to get the Web request GUID for each request. These ISAPI extensions also use the ISAPI API's to communicate with IIS. ASP.net is an ISAPI extension that calls into IIS via ISAPI API's to get the Web request GUID. The ISAPI extension can call in to ask for more of the server variables for the Web request.

[0046] Implementations provide a server support function to communicate with an application infrastructure level (e.g., middleware layer, ISAPI extension, etc.) to pass up the Web request GUID should the application infrastructure level wish to trace itself. Because the tracing infrastructure is made public, any desirable degree tracing can be performed. If the application infrastructure level is to be correlated with the trace log, then the server support function can be provided to make information in the trace log available. The server support function provides a means by which an ISAPI extension can call into a Web server to find out what the Web request GUID is for a particular Web request so that the ISAPI extension can log trace messages that can then be correlated back to the trace messages of the Web server. The server support function can be integrated with both ASP and ASP.NET. An example of a trace file for this environment is seen in the Appendix.

[0047] A report can be made on the trace file using the GUID for a Web request. An application that generates the report puts the tracings in the trace file back together to reconstruct what events happened during a process. The GUID is used by the report

generation application to locate relevant trace messages that can be strung together so as to represent data necessary to show a back tracking of everything (or only certain events of interest) that happened for a specific request. There can be, however, more that one Web request that is contained in the report. For instance, Web requests can be generated by use of a Web page. The Web page can have multiple Web requests (e.g., multiple functionalities). In a Web page associated with an e-commerce application, several different Web requests could be generated: a request to go to an initial Web page of multiple Web pages where an order can be placed; a request to place an order; a request to query the order; a request to change the order; a request to cancel the order; etc. The use of a Web request GUID enables all trace events/messages to be grouped together for a specific Web request.

[0048] Figure 3 depicts a flowchart for an exemplary process 300. Process 300, which illustrates an exemplary operation of the Web request event tracing tool implementations described herein, can be performed in the exemplary network environment 100 seen in Figure 1. A server, such as server 134 seen in Figure 1, can service a Web request while using process 300 so as to log events that happen into one or more event trace output files, such as are seen in Figure 2, during the servicing of the Web request.

[0049] Process 300 has a block 302 at which an export of a callback is made to a tracing infrastructure of the server. As mentioned above, exemplary tracing infrastructures are seen in Figure 1 as being Web request tracing application 140 and trace infrastructure 144 that is part of the operating system 142 of server 134. By way of example of block 302 in a WINDOWS® operating system provided by the Microsoft Corporation, when a Web process starts up via a start up routine, the Web process calls ETW to register which providers are to be listened for to determined whether they will request tracing. As such, block 302 registers a 'callback' connection to communicate that it has tracing available and

is seek interested providers that will request use of the Web request event tracing tool. At block 304, a command can be received to configure, start, or stop Web request tracing. Block 304 communicates with Web processes via a call back function that a request has been made by an entity for event tracing.

[0050] At a block 306, configuration of the kind of event tracing that is to be done is specified. The configuration allows the event tracing infrastructure to enable tracing and to communicate with a Web process for the particular kinds of events that are to be traced for a Web request. For instance, the requested events that are to be traced incident to a Web request can include the tracing of events related to one or more URLs, one or more filters, one or more CGIs, interactions with one or more static files, any kind of compression, events having to do with authentications or security, etc. At block 306, the level of verbosity to be reported for an event of interest can also be configured.

[0051] After the configuration at block 306, process 300 moves to a query 308. Query 308 is looped until it determines that an event of interest has occurred pertaining to a Web request, such as an event that happens at a server that is servicing the Web request. When so determined, process 300 moves to a block 310 at which a Web process within which the event of interest has occurred, incident to the serving of a Web request, communicates a trace message along with the configured verbosity level data to the tracing infrastructure. A component of the tracing infrastructure that receives the tracing messages can be, for instance, a kernel trace session component of the operating system of the server. Here, the kernel trace session can perform event buffering as trace messages that are received are processed.

[0052] At a block 312, in the depicted implementation, the tracing infrastructure associates a Web Request GUID with the trace message. The event is then logged by the tracing

infrastructure at block 314 into a trace output file (e.g., event trace log) along with the Web Request GUID of the corresponding Web request, including an event GUID for each event that is being traced. Note the Web Request GUID corresponds to the request and the event GUID corresponds to a particular event. The entry for the event includes a trace message that reflects the configured verbosity level for the event. The data in the entry can be in a binary format. Process 300 then loops back to block 308 to monitor the processing of the Web request for the occurrence of any more events of interest that might occur. After the Web request has been serviced by the server, process 300 can terminate (not shown).

[0053] An exemplary process 400 for reporting on the events that occur when a Web request is serviced is depicted by way of a flowchart in Figure 4. At a block 402, a report generation application (e.g., a consumer application) makes a demand to access an event trace output file. To obtain the events in the event trace output file of interest, the application uses a Web request GUID that corresponds to the Web request of interest. Alternatively, the consumer application can sort the event trace output file by Web request GUID first and then by timestamp. As such, a report can be generated from the sorted output to show those events that are grouped by Web request GUID. As a further alternative, the start and end events can be reported from the sorted output for a given event GUID, such as the start of a particular filter event and the end of the particular filter event. Of course, other reports can be generated from the traced events in the event trace log file.

[0054] A block 404 reads a trace message from the event trace output file and a query 406 determines whether there are no more trace messages in the event trace output file. If so, the process 400 moves to a block 414. Otherwise, a query 408 determines if the Web request GUID of interest matches the Web request GUID in the trace message. If not, the process 400 returns to block 404. Otherwise, at a block 410, the report generation

application uses each event GUID in the event trace output file that corresponds to the Web request GUID to generate a report the corresponds to the Web request GUID. Each event GUID can be used to map the binary format event data in the trace message into an event description that is in a format that is human readable. Here, a Managed Object Format (MOF) can be used. A MOF is a way to describe the event structure of data related to an event that has been logged as a trace message. The MOF maps the event GUID into a format that is human readable in order to translate from the original binary format. The event GUID for the trace event can be, for instance, the first 128 bits of the data corresponding to every event. In one implementation, Data Type (1) 208 can be an event GUID for an event 204 (k), as seen in Figure 2. Thus, the event GUID is used to 'decode' the binary data in a trace message.

[0055] At a block 412, the mapped readable event description is written to a storage location and process 400 returns to block 404. When process 400 moves to block 414, the storage location is accessed to retrieve one or more mapped readable event descriptions. Each retrieved event description can be output in a report.

[0056] Diagnostic personnel can review the report to determine what events happened when a Web request corresponding to the Web request GUID specified was being serviced by a server. The report can be useful in diagnosing problematic issues that arise on servers and clients, such as the performance of a Web request and/or issues relating to a failed Web request. Latency problems on a Web request, problems with particular installed filters, and a Web request that stops being serviced can also be analyzed through use of the report. The diagnostic personnel can perform analysis of Web requests using the report which can be desirably designed to contain only relevant information. As such, the report will be

significantly less verbose than conventional process dump files generated during execution of code for servicing the Web request.

[0057] A Computer System

Fig. 5 shows an exemplary computer system, such as a server, that can be used to implement the processes described herein. Computer 542 includes one or more processors or processing units 544, a system memory 546, and a bus 548 that couples various system components including the system memory 546 to processors 544. The bus 548 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory 546 includes read only memory (ROM) 550 and random access memory (RAM) 552. A basic input/output system (BIOS) 554, containing the basic routines that help to transfer information between elements within computer 542, such as during start-up, is stored in ROM 550.

[0058] Computer 542 further includes a hard disk drive 556 for reading from and writing to a hard disk (not shown), a magnetic disk drive 558 for reading from and writing to a removable magnetic disk 560, and an optical disk drive 562 for reading from or writing to a removable optical disk 564 such as a CD ROM or other optical media. The hard disk drive 556, magnetic disk drive 558, and optical disk drive 562 are connected to the bus 548 by an SCSI interface 566 or some other appropriate interface. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for computer 542. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 560 and a removable optical disk 564, it should be appreciated by those skilled in the art that other types of computer-readable media which can store data that is accessible by a computer,

such as magnetic cassettes, flash memory cards, digital video disks, random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the exemplary operating environment.

[0059] A number of program modules may be stored on the hard disk 556, magnetic disk 560, optical disk 564, ROM 550, or RAM 552, including an operating system 570, one or more application programs 572 (such as the Web request trace application 140), cache/other modules 574, and program data 576. The operating system 570 can include a Web request event tracing tool as described herein (such as the trace infrastructure 144). A user may enter commands and information into computer 542 through input devices such as a keyboard 578 and a pointing device 580. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to the processing unit 544 through an interface 582 that is coupled to the bus 548. A monitor 584 or other type of display device is also connected to the bus 548 via an interface, such as a video adapter 586. In addition to the monitor, personal computers typically include other peripheral output devices (not shown) such as speakers and printers.

[0060] Computer 542 commonly operates in a networked environment using logical connections to one or more remote computers, such as a remote computer 588. The remote computer 588 may be a personal computer, another server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 542. The logical connections depicted in Fig. 5 include a local area network (LAN) 590 and a wide area network (WAN) 592. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

[0061] When used in a LAN networking environment, computer 542 is connected to the local network through a network interface or adapter 594. When used in a WAN networking environment, computer 542 typically includes a modem 596 or other means for establishing communications over the wide area network 592, such as the Internet. The modem 596, which may be internal or external, is connected to the bus 548 via a serial port interface 568. In a networked environment, program modules depicted relative to the personal computer 542, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0062] Generally, the data processors of computer 542 are programmed by means of instructions stored at different times in the various computer-readable storage media of the computer. Programs and operating systems are typically distributed, for example, on floppy disks or CD-ROMs. From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory. The invention described herein includes these and other various types of computer-readable storage media when such media contain instructions or programs for implementing the blocks described below in conjunction with a microprocessor or other data processor. The invention also includes the computer itself when programmed according to the methods and techniques described herein.

[0063] For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

[0064] Any of the functions described herein can be implemented using software, firmware (e.g., fixed logic circuitry), manual processing, or a combination of these implementations. The term "logic" or "module" as used herein generally represents software, firmware, or a combination of software and firmware. For instance, in the case of a software implementation, the term "logic" or "module" represents program code that performs specified tasks when executed on a processing device or devices (e.g., CPU or CPUs). The program code can be stored in one or more computer readable memory devices. The illustrated separation of logic and modules into distinct units may reflect an actual physical grouping and allocation of such software and/or hardware, or can correspond to a conceptual allocation of different tasks performed by a single software program and/or hardware unit. The illustrated logic and modules can be located at a single site (e.g., as implemented by a single processing device), or can be distributed over plural locations.

[0065] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.